a) O algoritmo fica mais ilegível.
b) O algoritmo fica mais difícil de ser depurado.

Exemplo:

```
A ← 14, 2; I ← 1; enquanto I < 10 faça X ← X + 1; K ← I * K; I ← I + 1; fim enquanto;
```

O mesmo exemplo com cada comando em uma linha:

```
A ← 14, 2;
I ← 1;
enquanto I < 10 faça
|   X ← X + 1;
|   K ← I * K;
|   I ← I + 1;
fim enquanto;
```

Caso desejássemos incluir um novo comando dentro do *enquanto*, no primeiro caso será necessário reescrever toda a linha.

8. *Utilize parênteses para aumentar a legibilidade e prevenir-se contra erros.* Exemplo:

com poucos parênteses	com parênteses extras
A * B * C / (D * E * F)	(A * B * C) / (D * E * F)
A * B / C * D / E * F	((((A * B)/(C) * D)/E) * F
A ** B ** C	(A ** B) ** C
A / B / C / D	((A / B) / C) / D
X > Y *ou* Q	(X > Y) *ou* Q
A + B < C	(A + B) < C

9. *Utilize "identação" para mostrar a estrutura lógica do algoritmo.* A identação não deve ser feita de forma caótica, mas segundo certos padrões estabelecidos.

10. *Lembre-se: toda vez que for feita uma modificação no algoritmo, os comentários associados devem ser alterados, e não apenas os comandos.* Antes não comentar do que deixar um comentário errado.

Exemplos de trechos de algoritmos mal escritos e a análise das regras de programação violadas serão apresentados a seguir. Em alguns casos, o algoritmo equivalente, com qualidade, será apresentado.

a)
```
SOMA ← 0;                  {Atribui o valor 0 à variável SOMA}
I ← 1;                     {Atribui o valor 1 à variável I}
enquanto I < 18 faça       {Enquanto I não ultrapassar o valor 18, será atribuído o
|   SOMA ← SOMA + I;        valor da expressão SOMA + I à variável SOMA e o valor de
|   I ← I + 1;              I será incrementado de 1}
fim enquanto;
imprima (SOMA);            {Imprime o valor de SOMA}
```


Fere principalmente a regra nº 3. Ele deveria ser escrito para ser considerado "de qualidade" da seguinte maneira:

```
início   {faz a soma dos números inteiros de 1 a 18
          autor: EU – data: 01/01/81}
|  inteiro: SOMA;               {é o somatório}
|           I   ;               {número inteiro gerado}
|  SOMA ← 0;
|  I ← 0;
|  enquanto I < 18 faça         {gera os números inteiros e calcula o somatório}
|  |   SOMA ← SOMA + I;
|  |   I ← I + 1;
|  fim enquanto;
fim.
```

b)
```
início
    inteiro: XPT, I, II, III, IIIII;
    leia (XPT, IIIII);
    I ← 1;
    enquanto I < XPT faça
    se I < IIIII então se IIIII = 20 então II ← XPT + 2;
    senão III ← IIIII ** XPT; senão III ← XPT;
    IIIII ← III + I; fim se; fim se; {sem comentários}
    I ← I + 1;
    fim enquanto; Imprima (I, II, IIIII, III, XPT); fim.
```

Fere *todas* as regras sendo que as mais graves são: a falta de identação, nomes não significativos e ausência de comentários. Sugerimos ao aluno reescrever o algoritmo de forma a melhorar a qualidade.

3.2 METODOLOGIA DE DESENVOLVIMENTO DE ALGORITMOS

Uma das dificuldades naturais de um iniciante em programação é como começar a desenvolver um algoritmo para resolver um dado problema. Os passos seguintes, se seguidos, podem auxiliar nesta tarefa:

passo 1: *leia* cuidadosamente a especificação do problema *até o final.*
(fazer anotações)
ENTENDEU ← *falso*; VEZES ← 0;